METHOD AND APPARATUS FOR CALCULATING GEOMETRY OF A MOVING HAVEN

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention pertains to the field of marine navigation and more particularly to the calculation of the geometry of a moving haven along a navigation path.

2. Description of the Prior Art

A moving haven is a mechanism used in marine navigation to manage the voyages of marine vessels. A moving haven is generally a three-dimensional region which moves along a predefined path, a voyage plan. A vessel using a moving haven for navigation may move freely within the region, but may not go outside its boundaries.

The calculation of the geometry of the moving haven presents some difficult challenges due to the subtle variations of its shape for small changes in the parameters that define it. Prior solutions focused on drawing the moving haven in its more common manifestations and required special case logic to handle variations from the norm. Special case logic makes software more complicated, and it is difficult to make complex software work correctly. Furthermore, it is difficult to identify all the special cases, so some are inevitably missed.

A moving haven is a two dimensional region that moves through the ocean. A ship may not cross outside the moving haven boundary, but is allowed to operate within it freely.

A moving haven, in its simplest manifestation, is a rectangle. As it progresses through the water, it follows a predefined path, which is called a voyage plan, defined by a series of waypoints connected by legs. A rectangular buffer, the sides of which are a predetermined distance from the corresponding sides of the rectangle, provides an early warning to the navigator that the ship is approaching the edge of the moving haven.

As the name implies, the moving haven is in motion. When it approaches a waypoint it must turn from one leg to the next. When a rectangular moving haven spans a waypoint, the region that the moving haven encompasses generally includes the area of two overlapping rectangles and one pie slice shaped region, as shown in Figure P1. The first rectangle is defined around the line segment starting at the intersection of the trailing edge and voyage plan leg AB and ending at waypoint B. The second rectangle is defined around the line segment starting at waypoint B and ending at the intersection of the leading edge and the voyage plan leg BC. The pie slice shaped region, having an arc center at B, fills in the gap between the two rectangles.

Determining the geometry for the moving haven boundary must be accompanied by the determination of the geometry of the moving haven buffer. The moving haven buffer must be drawn so that each point on the buffer is exactly a predetermined distance from at least one point on the moving haven boundary, and no closer than the predetermined distance to any point on the moving haven boundary. It would be expected that the boundary has the same shape of the moving haven, but drawn smaller. This, however, is not the case. Often, when the moving haven has a sharp turn, the buffer requires a smooth curve, as shown in Figure P2.

Calculating the geometry of the moving haven boundary and its buffer is a difficult problem that has challenged engineers for years. The difficulties are a result of A) the moving haven often taking form of a variety of shapes and B) the fact that the buffer and boundary are not proportional, e.g. the buffer is, not simply a smaller rendition of the boundary.

Past attempts to determine the moving haven geometry have resulted in only marginal success, partially do to the following:

A) The problem was not decomposed adequately into smaller more manageable sub-problems.

B) Special case logic was required to handle irregular geometries.

C) No single solution was found to solve both the boundary problem and the buffer problem.

Past solutions have constructed the moving haven geometry sequentially, starting at some vertex on the boundary (or buffer) and working either clockwise or counter clockwise, determining subsequent vertices in the order that they appear in the final geometry. Such solutions also attempted to construct the boundary directly from the voyage plan geometry, rather than constructing intermediate results, and then making further refinements to arrive at the final geometry. The following is an example of a method that calculates the geometry sequentially.

Refer to Figure P3, assume P is the intersection of the trailing edge and the first leg, Q is the waypoint that the moving haven spans, and R is the intersection of the leading edge and the second leg. The moving haven boundary is shown with dark solid lines. The buffer is shown with a dotted line.

1) Point A is found by adding to point P a vector perpendicular to PQ of length equal to half the width of the moving haven.

2) Point B is found by adding to point Q a vector perpendicular to PQ of length equal to half the width of the moving haven.

3) Points are calculated to form an arc between B and C. The arc has a radius equal to half the with of the moving haven.

4) Point C is found by adding to point Q a vector perpendicular to QR of length equal to the half width of the moving haven.

5) Point D and E and found in a similar fashion.

6) A temporary line EH is found by shifting QR in a direction perpendicular to QR a distance equal to half of the moving haven width. A temporary line GI is found by shifting PQ in a direction perpendicular to PQ a distance equal to the half of the moving haven width. Point F is the intersection of these two lines.

7) Point G is found in a way similar to point A.

Notice that the steps outlined above are very specific to calculating a moving haven spanning a single waypoint as shown in Figure 5. Because the steps are specific to this particular scenario, a modification to these steps is required if the scenario changes in subtle ways. Furthermore, additional logic is required to detect that the scenario has changed. The following illustrates a few alternate scenarios and the difficulties these variations introduce.

As with the moving haven in Figure P3, the moving haven in Figure P4 spans a single waypoint. But this scenario differs in that the leading edge has just past a waypoint. As a result, the leading edge is not drawn in full. A portion of the leading edge is clipped by the area of the moving haven drawn around the first leg. Because the steps outlined above provide no provision for the leading edge being clipped, special case logic is required to detect this situation and account for its differences appropriately.

Figure P5 shows yet another scenario that requires special case logic. In this case, the leading edge is partitioned into two sections, because it intersects a portion of the moving haven that spans the first leg.

The above examples demonstrate the need for special case logic when the moving haven spans a single waypoint. It may be possible to identify and handle all special cases for the single waypoint situation, but the number of special cases becomes unmanageable when the moving haven spans two or more waypoints. As shown in Figure P6, the moving haven geometry can become quite complex when spanning as few as two waypoints. Determining the special case logic to calculate the boundary geometry for this shape directly from the voyage plan is difficult and likely to be error prone.

The above discussion focused on the difficulties in drawing the moving haven boundary. Calculating the moving haven buffer suffers from these and additional difficulties, making buffer problem harder to solve.

Prior art software fails, in many situations, to calculate the boundary and buffer geometries for the following reasons:

A) Prior art solutions did not adequately divide the problem into small simple sub-solutions, attempting to solve the entire problem at once. These programs attempted to calculate the final geometry directly from the voyage plan geometry.

B) Special case logic was required for handling variations in the voyage plan geometry.

C) Two separate methods were required for calculating the boundary and the buffer, resulting in more opportunities for errors to be introduced.

SUMMARY OF THE INVENTION

In accordance with the present invention, a process utilizing simple shapes, such as rectangles and arcs, is employed for calculating the moving haven geometry. This process builds the complicated moving haven geometry with the shapes and succeeds where the prior art failed because:

A) It decomposes the problem into smaller simpler sub-problems and

B) The approach is designed to handle all moving haven manifestations without special case logic.

The simple shapes method has the following characteristics:

A. The problem is subdivided into smaller,simpler sub-problems.

B. No special case logic is required for the various manifestations of the moving haven. In other words, the same sequence of steps is used regardless of the geometry.

C. The method used to calculate the moving haven boundary is also used to calculate the buffer.

The desired end result is an ordered list of points that define the vertices of the moving haven boundary. This is accomplished by providing software that is free

to subdivide the original problem into simpler sub-problems. Instead of focusing on the determination of the line that forms the boundary (and buffer) of the moving haven, the simple shapes method constructs a composite region that the moving haven defines . The region is always the union of one or more rectangles and zero or more pie slice shaped areas. This leads to the first two sub problems that must be solved:

A) Calculate the rectangles around each line segment of the input polygonal line. For calculating the moving haven boundary, the input polygonal line is the portion of the voyage plan spanned by the moving haven.

B) Calculate the pie slice shaped areas that complete the definition of the moving haven.

C) Find the boundary of the region that forms the union of the set of intersecting rectangles and pie slice shaped regions.

The solution of these sub-problems is relatively straightforward, the details of which will be discussed subsequently.

Various subtleties must be addressed to make the method work in practice. These three steps, however, form the essence of the method. Steps A and B are relatively simple. Step C is somewhat more involved.

The invention will be better understood by the description of the preferred embodiments with reference to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure P1 is an illustration of a moving haven split into multiple smaller rectangles.

Figure P2 is a comparison of a moving haven boundary and its buffer.

Figure P3 is an illustration of a moving haven spanning a vertex.

Figure P4 is an illustration of a moving haven just starting on a second leg.

Figure P5 is an illustration of a moving haven with its leading edge partitioned into two separate line segments.

Figure P6 is an illustration of a moving haven spanning two waypoints.

Figure 1A is an illustration of a moving haven spanning a waypoint useful for explaining its decomposition into simple shapes.

Figure 1B is an illustration of a rectangle drawn along the first leg spanned by the moving haven of Figure 1A.

Figure 1C is an illustration of a rectangle drawn along the second leg of the moving haven of Figure 1A.

Figure 1D is a combination of the basic shapes that form the moving haven of Figure 1A.

Figure 2A is a representation of rectangles and arcs generated for establishing a moving haven buffer.

Figure 2B is an illustration of a buffer formed by the rectangles and arcs of Figure 2A.

Figure 3 is a flow chart for determining a moving haven

Figure 4A is a representation of a portion of voyage plan about which a moving haven is to be drawn.

Figure 4B is representation of rectangles drawn about the portion of the voyage plan of Figure 4A.

Figure 5A is an illustration of arc generations for a moving haven boundary.

Figure 5B is an illustration of a line segment approximation of an arc.

Figure 6 is an illustration of a process for determining the arc segments of Figure 5B.

Figure 7 is a vector diagram useful for explaining the start line determination step.

Figure 8A is a set of vectors useful for explaining the creation of a polygonal line.

Figure 8B is a vector diagram useful for explaining the determination of counter-clock wise and clock wise turns.

Figure 9 is a set of line segments useful for explaining a method for determining a start segment for generating a buffer boundary.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the figures elements mentioned in a figure and appear in subsequent figures bare the same reference numerals.

Refer now to Figure 1A, wherein a moving haven is shown to proceed along a first leg 11 to a waypoint 13 and then along a second leg 15, forming a boundary 17. To calculate the moving haven a first rectangle 19 is drawn as shown in Figure 1B. This rectangle has a width W equal to the width of the moving haven and a length L_1 which starts at the trailing edge 21 of the moving haven and extends to the way point 13. A second rectangle 23 is drawn as shown Figure 1C. This rectangle has the width W and a length L_2 which starts at the waypoint 13 and extends to the leading edge 25. The rectangles 19 and 23 respectively positioned along the first and second legs establish a gap 27, shown in Figure 1D, on the boundary of the moving haven. This boundary gap 27 is closed by drawing an arc 29 from the corner 19a of the rectangle 19 to the corner 23a of the rectangle 23, thus forming a pie slice having sides 29 a and 29b. The boundary 17, shown in Figure 1A, of the moving haven comprises the sides 19b and 19c of the rectangle 19, sides 23b and 23c of the rectangle 23, sections 19d and 23d of sides of the rectangles 19 and 23, which respectively extend from intersection 33 to the sides 19c and 23c, and the arc 29.

Two of the three criteria identified as requirements for success have been discussed. First, the problem is subdivided into smaller simpler sub-problems, Second, the problem avoids special case logic in that the orientation and relationship of the intermediate shapes is inconsequential since any arbitrary combination of

adjoining rectangles and pie slice shaped areas may be combined.

As mentioned above, the method should be general enough to solve the buffer geometry problem as well. The polygon of the buffer may be determined if the method's input parameters are changed. For the moving haven, the input parameters include the portion of the voyage plan spanned by the moving haven and a value specifying the moving haven width. To determine the interior buffer, the input polygon is the polygon that defines the moving haven boundary, rather than a portion of the voyage plan. The value used for the width of the rectangle that is moved along the boundary is twice the buffer width, rather than the moving haven width.

As previously stated, the first step of the process is to generate rectangles around each line segment and pie slice shaped areas for each vertex. The application of this step results in the rectangles and pie slice shaped regions shown in Figure 2A. A first rectangle 33 is centered on the moving haven boundary, having a width that is twice the width of the buffer and a length equal to the distance between a first corner 35 of the moving haven boundary 39 and the start of an arc section 37. As will be explained subsequently, an internal arc 41a and external arc 41b are drawn for the duration of boundary arc section 37 and a second rectangle 43 is drawn from the end of the arc section 37 to a second corner 45 of the moving haven boundary 39. A third rectangle 47 is drawn between the second corner 45 and a third corner 49 of the moving haven boundary 39 and an arc 51 is drawn as previously discussed. This process continues about the boundary of the moving haven until the entire moving haven boundary has been traversed. The composite region drawn at the completion of the buffer geometry has interior and exterior edges. Since the buffer may not extend beyond the moving haven the interior edge 53, shown in Figure 2B wherein the result of the process is shown, is the edge of interest.

Refer now to Figure 3, wherein a data flow diagram for the simple shapes process is shown. An ordered list of points representing a polygonal line (a portion

of the voyage plan spanned by the moving haven) is assembled 55 and the width of the moving haven 57 are coupled to a rectangle generator 59 and an arc generator 61. The rectangle generator 59 provides sets of directed line segments representing rectangles, each set establishing a rectangle having the width of the moving haven around each of the line segments of the polygonal line. Refer now to Figures 4A and 4B. Consider a portion of a voyage plan having an origin 63a, two waypoints 63b and 63c, and an end point 63d. A first rectangle 65 is drawn between the origin 63a and the first waypoint 63b, a second rectangle 67 is drawn between the waypoints 63b and 63c, and a third rectangle 69 is drawn between the second waypoint 63c and the end point 63d. A first vertex 65a of the rectangle 65 is found by offsetting a distance of one half the moving haven width from the origin 63a, in a direction perpendicular to the segment between the origin 63a and the first waypoint 63b. The remaining vertices for the rectangle 65 are found in a similar fashion. The vertices of the rectangles 67 and 69 are determined in a same manner as the vertices of rectangle 65, the distances being measured on lines drawn perpendicular to the line segments between the waypoints 63b and 63c and the waypoint 63c and the end point 63d, respectively.

Line segments provided at the output 59a of the rectangle generator 59 must be constructed properly for use when the moving haven is established, which will be discussed subsequently. Establishment of the moving haven requires a set of directed line segments. The order of the line segments within the set is not important, but the direction of the line segments is critical. Each line segment must be constructed so that the line segments that define a rectangle, point in a clockwise direction. For example, the arrows which show the direction of the line segments for the rectangle 65 are collectively pointing in a clockwise direction. The same is true for the rectangles 67 and 69.

Refer to Figures 3, 4A, and 5A. The ordered list of points representing the polygonal line from the polygonal line assembly 55 and the width provided from moving haven width 57 are coupled to arc generator 61 wherein arcs are generated

with a specified radius for each point on the input polygonal line requiring an arc segment. At each of these points, the arc is generated around the obtuse angle formed by the intersecting line segments. To complete the moving haven for the polygonal line shown in Figure 4A, an arc 71a must extend from the vertex 65b of rectangle 65 to the vertex 67a of the rectangle 67 and an arc 71b must extend from the vertex 67d of the rectangle 67 to the vertex 69c of the rectangle 69.

These arcs are approximated by a series of directed line segments, as shown in Figure 5B three directed lines are chosen to approximate the arc, a line segment 72a from point 73a to point 73b, a line segment 72b from point 73b to point 73c, and a line segment 72c from point 73c to point 73d. Just as with the generation of rectangles, the line segments representing the arc must be generated in a clockwise direction.

In the above illustration, line segments 72a, 72b, and 72c approximate the arc. These line segments form a clockwise rotation along the circle that defines the arc. Though only three line segments were used to illustrate the arc approximation, it should be understood that a greater number of line segments may be used to yield a smoother curve.

The following steps may be applied to approximate of an arc for an input polygonal line ABC, shown in Figure 6.

1 .A vector X_1 that connects the vertex to the beginning of the arc is determined. This will be a vector with a length equal to the specified radius, starting from vertex B, and perpendicular to AB.

2. A vector X_2 that connects the vertex B to the end of the arc is determined. This will be a vector with a length equal to the specified radius, starting from vertex B, and perpendicular to BC.

3. Add the point R_1 at the end of vector X_1 to an ordered list of points.

4. Rotate X_1 by an angle θ to establish a vector X_3. Smaller values of θ will result in a better approximation of the arc. If the total angle between the vector X_3 and the start vector X_1 is less than the angle

between the end vector and the start vector, add the point at the end of the vector X_3 to the ordered list.

5. If the total angle between X_3 and the start vector X_1 is less than the angle between the end vector and the start vector, repeat step 4 rotating the vector X_3 through the angle θ and adding the point at end of the rotated vector to the list. Otherwise proceed to step 6.

6. Add the point R_3 at the end of the end vector X_2 to the list.

If θ was set to 30 degrees for the arc shown in Figure 6, the process would generate the 4 points (73a, 73b, 73c, and 73d) shown in Figure 5B to define the arc shown in Figure 6. The final output of this step of the method would be the line segments 72a, 72b, and 72c which connect these points.

Arc representative line segments provided at the output 61a of the arc generator 61 are vectors, like the vectors representative of the rectangles, are oriented in a clockwise direction. This set of line segments and the set of line segments for the formation of the rectangles are combined in a combine sets step. 75. The resulting set of combined line segments sets are provided at output 75a of set combiner 75 and coupled therefrom to a start line segment determinator 77.

Before the boundary of the moving haven can be established by a moving haven boundary generator 79, a starting line segment must be specified. The boundary determinator 79 receives the combined line segments from the output 75a of the combined sets step 75 and a starting line designated by the start line determinator 77. Establishment of a correct moving haven boundary requires a starting line segment which is on and has its beginning on the moving haven boundary. Referring to Figure 7, line segments 81a, 81b, 81c, having the entire segment on the boundary, and line segments 81d, 81e, having the beginnings on the boundary, are acceptable; while 81e 81f, 81g, 81h, 81i, although having portions and their ends on the boundary, their beginnings are within the boundary, and are therefore, not acceptable. Line segment 81j is not acceptable, it is entirely within the boundary. The segments that approximate the arcs 82 and 84 are all acceptable.

The following heuristic may be used to find an appropriate starting line segment:

I .Let R be the set of all segments that have start points that are at least as far left as all other points. Segments that meet this criteria would be 81a and 81b.

2. If R contains only one segment, then this is the start segment.

3. If R contains more than one segment, then select the line segment in R that points the most upward. Since segment 81a points more upward than 81b, it is selected.

The boundary generator 79 (Figure 3) requires that the intersection of two segments to be classified as either clockwise of counter-clockwise. Refer to Figure 8B, wherein clockwise and counter-clockwise turns are illustrated. A turn from vector 85 to 87a requires a counter-clock wise movement. The angle θ_1 from the vector 85 to the vector 87a is less than 180°, as is the angle θ_1', the angle from the vector 85 to the vector 87b, which also requires a counter-clock wise turn for the vector transition. It should be apparent that all counter-clock wise turns will have angles that are less than 180° between the transition vectors. A turn from vector 85 to vector 89a is a clockwise turn and the angle θ_2 from vector 85 to vector 89a is less than 180°, as the angle θ_2', the angle between the vector 85 and 89b, which requires a clock wise turn for the vector transition. It should also be apparent that all clock wise turns will have angles between transition vectors that are greater than 180°.

To illustrate the operation of the boundary generator 79 (Figure 3), refer to Figure 8A. Assume segment 83a, with a beginning point A and an ending point B, is a starting segment for a polygonal line. To complete the polygonal line the following procedure may be employed with reference to Figure 8A:

1. Add the beginning point A of the segment 83a to an ordered list L, which will be the list of points that define the final polygonal line.

2. Set the current segment 83a to be the starting segment. Call 83a X.

3. Look for intersections between X and any other segment. In Figure 8A, the segment X intersects 83b, 83c, 83d, 83e, and 83f. Call this list of segments the candidate set of segments, named S.

4. Drop all segments from the set S whose end point touches the starting point of the starting segment X. This eliminates segment 83e.

5. Drop all segments from the set S that result in a clockwise turn, unless that segment intersects X at the end point B. This eliminates 83b, but not 83f.

6. For each of the remaining line segments in S, find where those segments intersect X. Only segments 83c, 83d, and 83f remain in S. Call the point that at which 83c intersects X, point Q. Notice that 83d also intersects X at Q. Call the point that 83f intersects X, point P. Of the intersection points, find the one that is closest to the start point A of X. In Figure A, Q is closest to A. Drop all line segments that do not include this closest intersection point. This removes 83f from S, leaving only 83c and 83d.

7. If S is empty then terminate the process.

8. Of the remaining segments in S, select the one that results in the smallest angle θ, where θ is measured as shown in Figure 8B. Call the selected segment W. In Figure 8B, the angle between X and 83d is smaller than the angle between X and 83c, so W would be set to 83d.

9. Let Z be the intersection of W and X. In Figure 8B Z = Q. If Z is already in L, add Z to the end of the list L and terminate the process. Otherwise, add Z to the end of the list L, and go to the next step.

IO. Set X to be the segment that starts with Z and ends with the end point of W. Repeat, starting with step 2 and using segment 86, having its beginning point at point Q and its ending point at point C, as the starting segment.

Once the moving haven boundary has been determined the moving haven buffer is calculated.

An overview for calculating the moving haven buffer has been previously discussed. The primary differences in applying the simple shapes for the determination of the buffer and applying the simple shapes to the determination of the moving haven boundary. are: .

A) For determining the buffer, the input polygonal line is the geometry of the
moving haven boundary, rather than the voyage plan. The value used for the width is twice the buffer width, rather than the moving haven width.

B) The input polygon for arc generator 61 (Figure 3) must be altered. Specifically, a vertex must be added to the end of the input polygonal line that has the same coordinates of the second vertex of the moving haven boundary.

C) The rectangle generator only generates the portion of each buffer rectangle that is within the moving haven boundary. Also the arc generator only generates arcs within the boundary.

D) A different method must be used for determining the starting segment input to the boundary generator.

In determining the moving haven boundary, there is no need to draw an arc around the first or last vertex of the input polygon. Because the buffer is a closed polygon, it is necessary to draw an arc around each vertex. Rather than modifying the arc generator 61 (Figure 3) when creating the buffer, the input polygon is altered by adding a vertex to the end of the input polygonal line that has the same coordinates as the second vertex of the moving haven boundary. The addition of the second vertex of the moving haven boundary to the end of the input polygonal line and the actual end point of the boundary is no longer the last point of the input

polygonal line. As a consequence an appropriate arc will drawn for each vertex of the unaltered input polygonal line.

The buffer, by definition, will never extend outside the moving haven boundary. As a consequence, any portion of any segments created by the arc or rectangle generators that are outside the boundary will never be a part of the final buffer geometry. To reduce the number of segments that the boundary generator processes, the arc generator only creates segments for arcs that are within the moving haven boundary and the rectangle generator only creates the portion of the buffer rectangle that is within the moving haven boundary, as shown in Figures 2A and 2B and described in the text relating thereto.

The method for finding the start segment to be used as input for the boundary generator 79 (Figure 3) that was used for the moving haven boundary is not appropriate for the generation of the buffer. To establish the polygonal line for the buffer having segments 91a, 91b, 91c, and 91d, illustrated in Figure 9, the following sequence of steps may be performed:

1) Let R be the set of segments used as inputs into the boundary generator 79 (Figure 3). R includes the segments 91a, 91b, 91c, and 91d.

2) Find a segment in R that is as least as long as all other segments in R. Segment 91a meets this criteria.

3) Find the center of 91a and label it C, so that 91a comprises two segments 93a, terminating at C, and 93b, originating at C.

4) Replace segment 91a with segments 93a and 93b.

5) Use 93b as the starting segment for the buffer bounder generation in boundary generator 79.

6) Modify the set R so it includes segments 93b, 91b, 91c, 91d, and 93a.

While the invention has been described in its preferred embodiments, it is to understood that the words that have been used are words of description rather than

limitation and that changes may be made within the purview of the appended claims without departing from the true scope and spirit of the invention in its broader aspects.

5